Tema 1 CAVA - Mathable

Alexandru Marius-Cristian

3 decembrie 2024

Rezumat

1 Introducere

Scopul acestui proiect este dezvoltarea unui sistem automat pentru a calcula scorul ˆın jocul Mathable. Mathable este un joc bazat pe ecuații matematice care trebuie formate pe tabla de joc. A fost descris ca fiind similar cu jocul Scrabble, dar folosind numere.

Avem de rezolvat 3 cerințe:

1. **Detectarea** poziției piesei noi adăugate pe tablă după fiecare mutare
2. **Identificarea** numărului de pe piesa nou adăugată
3. **Calcularea** scorurilor obținute de fiecare jucător în runda sa

Soluția propusă este una în jupyter notebook și folosește următoarele module: numpy, opencv-python, os

2 Detectarea poziției

Prima sarcină implică identificarea poziției pe tablă unde a fost plasată o piesă nouă. Din imaginile cu tabla de joc pe masă trebuie să extragem tabla de joc la limitele ei, apoi să o împărțim în celulele componente și să analizăm interiorul celulei pentru a decide dacă conține un număr sau nu.

2.1 Procesarea imaginilor cu tabla de joc

Funcția extrage_careu are rolul de a identifica și extrage zona principală de interes din imaginea cu tabla de joc. Pentru a ușura procesul de identificare și extragere a zonei de interes din imagine am pefecționat câteva filtre de culori în spațiul HSV după modelul din laboratorul de filtrare al culorilor. Acestea sunt definite și se pot aplica cu funcția color_filter, și ofer mai jos un exemplu în Figura 1 și Figura 1. (vezi folderul ./myaux/)

Funcția aplică filtrul pentru detectarea colțurilor, apoi folosește metode OpenCV pentru preprocesarea imaginii prin dilatare pentru a solidica conturul , detectarea contururilor, identificarea colțurilor potențiale din care este ales grupul de arie maximă. Imaginii inițială este trecută în grayscale și îi se aplică o transformări de perspectivă pentru a alinia zona selectată într-o grilă aproximativ în formă de pătrat. Funcția returnează imaginea transformată, masca asociată și coordonatele colțurilor detectate.

2.2 Împărțirea pe celule și condiția de decizie

Imaginea transformată este apoi împărțită într-un grid 14x14. Asupra fiecărei celule (patch) rezultate se aplică un threshold binar care trece pixeli gri-deschis de fundal la pixeli albi. Rezultă apoi 3 tipuri de patchuri: albe goale, majoritar negre cu o modificare de logică de joc și cu numere scrise negru pe alb. Cele de interes sunt cele cu numere, a căror medie de intesitate a pixelilor se află în intervalul (120,245).

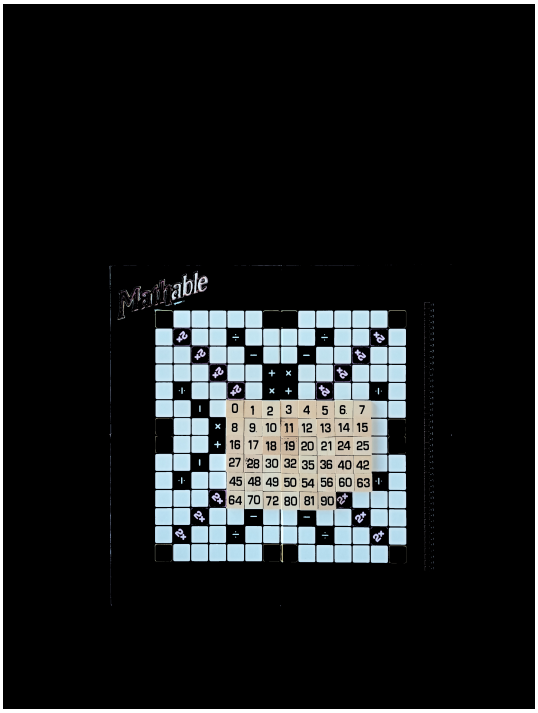

Figura 1: Rezultatul în urma aplicării filtrului de tip 'dark_table'

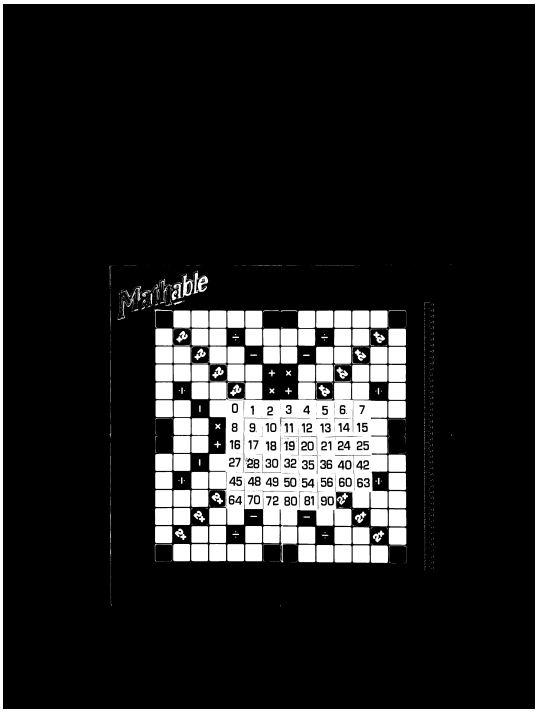

Figura 2: Masca folosită pentru filtrul de tip 'dark_table'

3 Identificarea numărului de pe piesă

3.1 Template Matching

Pentru clasificarea numerelor folosim metoda template matching. Avem nevoie însă mai întâi de niște templateuri robuste și centrate, precum și de funcții de preprocesare a patch-urilor. Din datele de antrenamre extragem pentru fiecare patch-ul nou și numărul de pe piesă, apoi centrăm patch-ul pe număr și îl salvăm într-un folder pentru valoarea numărului. Metoda de centrare presupune încadrarea cifrei într-o fereastră și calcularea centrului ferestrei. În cazul numerelor de 2 cifre vor fi 2 ferestre care trebuie apoi unite într-un grup mare de ferestre (vezi laboratorul 8). La fel procedăm cu numerele din imaginea auxiliară 3, apoi calculăm media templateurilor pentru fiecare număr. Aceste templateuri medii for servi drept templateuri de referință pentru metoda de clasificare template matching

3.2 Funcțiile de identificare propiu-zise

Similare cu funcțiile de detectare, cu excepția că se și clasifică cifra

4 Calcularea scorului

Putem hardcoda matrice ce să reprezinte tabla goală de joc și o mască de bonusuri și constrângeri din logica de joc.

4.1 Eval Score

Funcția de evaluare scor iterează prin rundele jucătorilor și prin mutările din fiecare rundă. Se aplică metodele de mai sus din care rezultă o matrice ox unde 'x' marchează prezența unui număr, o matrice fullboard în care se clasifică pentru fiecare imagine nouă fiecare piesă cu numere și o matrice current_game_board în care la fiecare mutare se clasifică piesa nouă adăugată și se scrie în matrice. Putem vedea poziția piesei noi prin identificarea diferenței între matricea ox curentă și matricea ox anterioară.

Încercăm apoi într-un bloc try să potrivim un număr din posibilele clase de clasificare în ordinea scorului de corelație rezultat de Template Matching într-o ecuație corectă pe tabla de joc. Uneori clasa de număr cu scorul de corelație cel mai mare nu este cea corectă și nu satisface o ecuație pe tablă, dar a treia sau a patra variantă după scorul de corelație va satisface ecuația și va fi foarte probabil clasificarea corectă.

În cazul în care nicio variantă nu satisface o ecuație pe tablă se va arunca o excepție IndexError. În acest caz intervine idea matricii fullboard. Am obervat prin 'trial and error' că unele piese pot fi clasificate diferit în imagini ulterioare, așa că încercăm să găsim unde a fost greșeala anterioară de clasificare și să o corectăm. Implementarea actuală greșește scorul în cazul în care anomalia este într-o rundă anterioară.

Pentru fiecare mutare se caută o ecuație validă în cele 4 direcții și scorul se calculează după regulile jocului.